REMARKS/ARGUMENTS

This paper responds to the Office Action of October 25, 2004, and requests reconsideration of the application. As noted in the accompanying Petition for Withdrawal of Finality, final rejection is premature, and this paper is entitled to entry as of right.

Claims 1-60, 63-115, and 118-135 are now pending, a total of 131 claims. Claims 1-19, 21-33, 37-47, 49-59, 61-85, 87-126 and 128-133 are not allowed; however as noted below, many of them are not rejected, either. Of the non-allowed claims, claims 22, 51, 63, 87, 94, 96 and 104 are independent.

The shortened statutory period is extended to run through April 25, 2005. Accordingly, this response is timely.

I. Entry of Amendments Under Rule 116

By the accompanying "Request for Withdrawal of Finality," Applicant submits that the finality of the Office Action of October 2004 was premature, and that the amendments proposed here may be entered as of right.

Even if finality is maintained, these amendments are entitled to entry under Rule 116.

The amendments to claim 104 respond to a new issue first raised in the Advisory Action of February 14, 2005. The amendments to claim 63 respond to an explanation first provided in the Office Action of October 2004. These amendments could not have been proposed earlier, because they could not respond to rejections that had not been raised.

The amendments to claims 87 and 92 are intended neither to clarify nor to change the scope of these claims, but rather to express the claims in language closer to that of the other claims, and more idiomatic to the art. Such changes reduce issues for appeal by reducing variability among the claims' language, put the claims in better form for consideration on appeal, and introduce no new issues.

Therefore, even if final rejection is maintained, these amendments may be entered under Rule 116.

II. Paragraphs 5-8: Incorporation by Reference

The material incorporated by reference into this application fully complies with all rules, including MPEP § 608.01(p)(A).

First, MPEP § 608.01(p)(B) states that there is no limit on material that may be incorporated from priority applications. As noted in the first sentence of this application, the applications incorporated by reference are priority applications, and thus under MPEP § 608.01(p)(B), any limitations on incorporation by reference "do not apply." Paragraph 7 of the Office Action has no legal basis.

Second, the Office Action reflects an incomplete reading of MPEP § 608.01(p)(A). § 608.01(p)(A) only limits incorporation of "essential material," and identifies "essential material" as "that which is necessary to (1) describe the claimed invention, (2) provide an enabling disclosure of the claimed invention, or (3) provide the best mode (35 U.S.C. 112)." § 608.01(p)(A) clarifies that the following are necessary elements of any objection:

- identify particular claims to which the material is "essential"
- identify the legal basis under which the material is essential – enablement, written description, best mode, or other
- identify the material that is "essential" to the identified claims

The Office Action never makes any averment that the material is "essential material" – if the material is not "essential," § 608.01(p)(A) states that such "nonessential subject matter may be incorporated by reference..." Further, without the three required showings, it is not at all clear that any requirement has any valid legal basis, and no applicant can make an informed decision on how to respond to a requirement. As a basic principle of administrative law, a requirement that does not permit a response has no legal existence.

Third, as the Office Action itself acknowledges, the material submitted on microfiche is not a "computer program listing." Therefore, new 37 C.F.R. § 1.96 does not constrain the form in which the material may be submitted. Further, the material submitted on microfiche exceeds the size of an appendix that may be submitted for printing under either old or new Rule 96, and it is not the type of information that may allowably be submitted on CD-ROM. In absence of any other appropriate medium, microfiche is an entirely proper way to place this material in the file for incorporation by reference unless and until it is shown to be "essential" to specific claims.

No incorporation by reference breaches any rule, past or present. Any objections may be withdrawn.

III. Claim 51

Claim 51 is discussed at ¶¶ 29 and 30 of the February Office Action, and ¶ 14.2 of the October Office Action, in the context of Goetz '913 alone. Claim 51 recites as follows:

> 51. A method, comprising:
>
> storing instructions in pages of a computer memory managed by a virtual memory manager, the instruction data of the pages being coded for execution by, respectively, computers of two different architectures and/or under two different execution conventions;
>
> in association with pages of the memory, storing corresponding indicator elements indicating the architecture or convention in which the instructions of the pages are to be executed, the pages' indicator elements being stored in a table whose entries are indexed by physical page frame number;
>
> executing instructions from the pages in a common processor, the processor designed, responsive to the page indicator elements, to execute instructions in the architecture or under the convention indicated by the indicator element corresponding to the instruction's page.

Claim 51 recites that the pages' indicator elements are stored in "a table whose entries are indexed by physical page frame number." In contrast, the Office Action points to Goetz '913, Fig. 4, and col. 10, lines 4-28, which read (in pertinent part) as follows (hand-written box and annotations added):

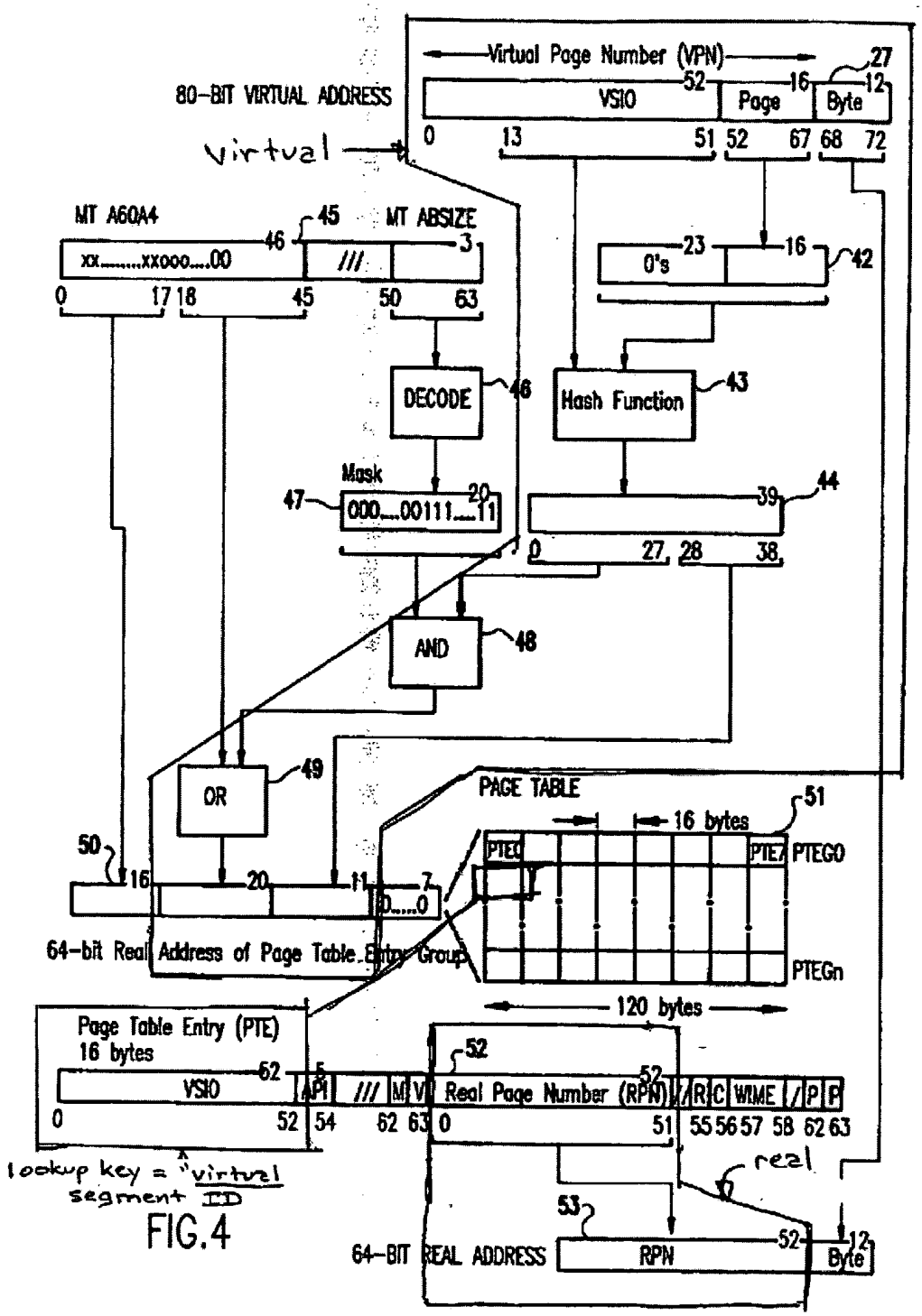


FIG.4

.... In the PowerPC architecture as shown in FIG. 4, VAs [virtual addresses] in register 27 are translated to physical addresses (PAs) via a hashed page table search. More particularly, bits 52 to 67 (page field) of the 80-bit virtual address in register 27 [are combined with other values] to form ... page table origin register 50. ... The highest seven bits of register 50 are forced to zero to form the real

> address of the page table 51 in memory. The individual page table group entries are searched until an entry 52 is found whose virtual segment ID matches that of the original VA. When found, the real page number is extracted from the page table group entry 52...

That is, the virtual address is used to determine the real address of the page table. Then the individual entries of the page table are searched based on the virtual segment ID to find a particular entry. Only after an individual page table entry is located based entirely on the virtual address of the address to be translated is a real address of the target page extracted from the entry.

This portion of Goetz '913 states four things that are incompatible with the interpretation expressed in the Office Action:

(a) Register 50 contains the real address of only the "page table" 51 as a whole. No real address is ever used to "index" an "entry," as recited in claim 51.

(b) To find a particular entry in the page table (as recited in claim 51) Goetz '913 uses various portions of the "virtual" address, not any value related to any "physical" address, as recited in claim 51.

(c) Goetz relies on the difference between "real" addresses (or "physical address," as recited in claim 51) and "virtual" addresses for correct behavior.

(d) Goetz '913 "extracts" the "real page number" from the page table entry that was located based on its "virtual segment ID."

The contrary view expressed in the Office Actions, that the "real address" of a page is used to "access a particular element" from which Goetz then "extracts" the real page number of the very page whose entry is indexed, makes no engineering sense. Using a real address to index a table to "extract" the very same real address is just silly. The scheme proposed in the October Office Action is essentially the same as looking in the white pages of the phone book to find the correct spelling of a person's last name (the real page number), after one already knows the correct spelling of the last name. No person would rationally do such a thing, or design such a step into a computer – clearly Goetz '913 does not teach the technique set out in the Office Action.

The October 2004 Office Action further confuses the issues by omitting the relevant portion of the Microsoft Computer Dictionary. In pertinent part, the relevant Microsoft definition of "index" reads as follows:

> **index** ... In programming, a scalar value that allows direct access into a multielement data structure such as an array. The index allows the programmer to ... derive the location of the desired element. ...

The "elements" of the second sentence of this definition are directly relevant to the "entries" recited in claim 51. Because this second sentence was entirely omitted from the Office Action, the Examiner's view with respect to the "elements" of the definition, or the "entries" of claim 51, cannot be discerned. Further, the first sentence uses the word "into," to clarify that "index" refers to access to the "elements" of the second sentence. Merely ascertaining the location of the "multielement data structure" in bulk, with no differentiation among the "elements," is not "indexing."[1]

Applicant notes that this explanation is essentially a repetition of the Response to Office Action of January 2005. However, the Advisory Action of February 2005 fails to "Answer All Material Traversed." Until the Examiner states a position that reflects the ordinary understanding of the terms "index," "entries," and "physical page frame number" and the very dictionary definition he relies upon, Applicant is unable to meaningfully respond further.

Claim 51 recites a limitation absent from Goetz '913, and is therefore patentable over Goetz '913.

IV. Claim 63

Claim 63 is discussed at ¶¶ 32 and 34 of the February 2004 Office Action, and ¶ 14.3 of the October 2004 Office Action, in the context of Goetz '913 alone. The February 2005 Office Action is silent with respect to the substantive issues of claim 63. Claim 63 recites as follows:

> 63. A microprocessor chip, comprising:
>
> an instruction unit, configured to fetch instructions from a memory managed by the virtual memory manager, and configured to execute instructions coded for first and second different computer architectures or coded to implement first and second different data storage conventions;
>
> the microprocessor chip being designed (a) to retrieve indicator elements stored in association with respective pages of the memory, each indicator element

[1] The current 5th edition of the Microsoft Computer Dictionary confirms that an "indexed search," using a physical page frame number as a search key, would be equivalent to "indexing" in the sense used in the older edition's definition, subject to the limits of the prior art. Goetz '913 raises no such issues, since anything in Goetz '913 relating to "indexing" of "entries" is based on virtual address.

indicating the architecture or convention in which the instructions of the page are to be executed, and (b) to recognize when instruction execution has flowed or transferred from a page of the first architecture or convention to a page of the second, as indicted by the respective associated indicator elements, and (c) to alter a processing mode of the instruction unit or a storage content of the memory to effect execution of instructions in accord with the indicator element associated with the page of the second architecture or convention;

wherein the indicator elements are stored in a table distinct from a primary address translation table and from portions of the primary address translation table cached in a TLB (translation lookaside buffer) used by a virtual memory manager, the indicator elements of the table being stored in association with respective pages of the memory.

Applicant submits that the "table" with "indicator elements" of the last paragraph of claim 63 distinguishes both Goetz' page table 51, 52 and TLB. Claim 63 is therefore patentable over Goetz '913.

V. Claim 87

Claim 87 is briefly mentioned in paragraph 100 of the Office Action of February 2004, in relation to some unspecified portions of Goetz '913, Brender '422, and Murphy '947, combined in some unspecified way. No language of claim 87 is compared to any reference, and thus no rejection was raised. The October 2004 Action, ¶ 14.4, compares an incorrect paraphrase of single limitation of claim 87 to Brender '422, col. 15, lines 48-50, but makes no comparison of claim 87 as a whole to any prior art. The February 2005 Advisory Action is silent in response to the substantive arguments raised in the Response to Office Action of January 2005.

Claim 87 recites as follows:

87. A method, comprising the steps of:

executing a control-transfer instruction under a first execution mode of a computer, the instruction being architecturally defined to transfer control directly to a destination instruction for execution in a second execution mode of the computer;

before executing the destination instruction, altering the data storage content of the computer to establish a program context under the second execution mode that is logically equivalent to the context of the computer as interpreted under the first execution mode, the reconfiguring including at least one data movement operation not included in the architectural definition of the control-transfer instruction.

A. No Office Action has Ever Addressed the Language of Claim 87 – No Rejection Exists

Neither Office Action has ever addressed the language of claim 87, an "instruction" with certain properties. Instead, the Office Action points to text in Brender '422 that discusses transferring control between "routines." No Office Action has never mentioned an "instruction" in the context of claim 87. Until an Office Action addresses the precise language of claim 87, no rejection exists.

B. The Technological Analysis of The October 2004 Office Action is Incorrect

The October Office Action misstates the content of Brender '422. Brender '422 makes clear that no "instruction" meeting claim 87 can exist.

First, Brender '422, col. 8, lines 31-37 and col. 10, line 43 through col. 15, line 16 teaches that his compiler identifies every transfer of control between different instruction sets, and then inserts a software "jacket" at that transfer. That "jacket" is interposed between the source "instruction" and destination of the transfer – there is never a "direct" transfer at an instruction level, as recited in claim 87.

Second, Brender '422 then absolutely ensures that there is no possibility of a direct transfer from a source instruction to the target of that instruction (across an instruction set difference), except through the jacket: before "Y" routines are linked with "X" routines ("X" and "Y" being the two instruction set architectures discussed in Brender '422), "The names of the called [Y] routines are hidden at link time..." (col. 15, lines 42-44). Because the names of the "Y" routines are hidden before they are linked with "X" routines, the linker cannot possibly link an "X" instruction to "transfer control directly to a destination instruction" in the "Y" instruction set. Brender '422 must do this to ensure that the software jacket is interpolated on every transfer between instruction sets.

Third, Brender '422 expressly teaches away from any "direct" transfer of control across an instruction set boundary, without going through the jacket. Such transfers are blocked as illegal. Brender '422 teaches that these instructions are architecturally defined to fault, not to "transfer control" as recited in claim 87. Paragraph 14.4 of the October Office Action quotes one of the clearest of these statements, Brender '422, col. 15, lines 48-50:

> In the present embodiment, it is based on the fact that a direct X call to a Y routine incurs an X operand fault.

This sentence from Brender '422 teaches opposite the interpretation suggested in the Office Action: an attempted "direct call" is architecturally defined to "fault," that is, to not complete at all. Such an instruction is not architecturally defined to "transfer control directly" to a destination, as recited in claim 87.

Because any correspondence between this language of claim 87 and any prior art rests on faulty technical analysis, claim 87 may be allowed.

C. Goetz '913 is Not Combinable with Brender '422 or Murphy '947

1. The Office Actions are Procedurally Incomplete, and Therefore Inadequate to Raise Any Rejection

The Director instructs, as a matter of "procedure," that the examiner bears the initial burden to come forward with three showings to support any rejection for obviousness:

> **2142 Legal Concept of *Prima Facie* Obviousness**
>
> The legal concept of *prima facie* obviousness is a procedural tool of examination which applies broadly to all arts. ... The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness. ...
>
> **ESTABLISHING A *PRIMA FACIE* CASE OF OBVIOUSNESS**
>
> To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.

The Examiner disagrees with the Director. No office Action has ever addressed "reasonable expectation of success" for any combination of Goetz '913 with either Brender '422 or Murphy '947.[2] Examiners do not have discretion to rewrite the rules – until an Office Action addresses all elements of *prima facie* obviousness, no obviousness rejection exists.

---

[2] In spite of several requests for some showing of "reasonable expectation of success," the Examiner refuses to provide any such showing. Indeed, in the paper of February 10, 2003, Examiner Ellis states that he believes that "reasonable expectation of success" only applies in the chemical arts, not "all arts" as stated in MPEP § 2142.

2. Goetz '913 is Technologically Incompatible with Brender '422 or Murphy '947

Goetz '913 uses an approach that is fundamentally incompatible with the approach of Brender '422 or Murphy '947. They address different problems in different ways, using incompatible techniques.

Goetz '913 teaches hardware that is capable of executing either a rough approximation of the Intel X86 (now Pentium) instruction set (a "complex instruction set computer," or "CISC"), and a modified IBM/Motorola PowerPC (a "reduced instruction set computer" or "RISC") instruction set (Goetz '913, Fig. 8). Goetz' overall approach is to modify one architecture or the other by dropping a feature here or adding a feature there to eliminate any data incompatibility between the two architectures. *E.g.*, Goetz '913, col. 18, lines 63-64 (X86 data format is "completely replaced" with a modified PowerPC format).

The only cases where Goetz '913 allows a difference to persist is where a switch between the two architectures can be implemented with no software intervention or change to state, beyond simply switching the instruction pipeline from X86 mode to PowerPC mode. For example, Goetz '913, col. 15, lines 26-29 discusses the difference between Intel X86 80-bit floating-point registers and PowerPC 64-bit floating point registers. No adjustment is required on transition. Goetz' CPU leaves all the data untouched: Goetz' 913 expressly states that floating point data is left in "the same register files," Goetz '913, *id.*, all he has to do is enable or disable 16 bits of the floating-point data path. No modification of data storage content is required for such differences.

In contrast, Brender '422 and Murphy '947 are directed to systems for emulating an Alpha (the 1990's RISC computer developed by Digital Equipment Corp.) on a VAX (Digital's 1980's CISC processor), and vice-versa. Neither Brender '422 nor Murphy '947 mention any specialized hardware to allow one computer to execute binaries for the other.

Both Brender '422 and Murphy '947 take similar approaches: both analyze the program to be executed to identify every conceivable point at which execution crosses from VAX binary code to Alpha binary code (in some cases, after binary translation from one instruction set to the other). The overwhelming bulk of both Brender's and Murphy's disclosures are directed to the various cases that may arise: how they are detected, and how they are handled. Based on that

analysis, Brender '422 and Murphy '947 both build a data structure for each and every case[3] of cross-architecture call that can be detected, Murphy '947, col. 6, lines 6-7 ("each incoming call"), col. 6, lines 14-15 ("outbound call"); Brender '422, col. 8, lines 32-34 ("each X [VAX] and Y [Alpha] routine"), col. 10, lines 50-53 ("Jacketing requires that knowledge of all the relevant calling convention characteristics for each subprogram in each domain be available to the Jacketing mechanism.").

There are three differences between Goetz '913 on the one hand, and Brender '422 and Murphy '947 on the other, that defeat any combination:

1. The "principle of operation" for Goetz '913 is to modify the two architectures – add some features here, remove some there – until any switch between them can be effected smoothly. In contrast, both Brender '422 and Murphy '947 must take both architectures without modification. For example, Brender '422 states that his goal is to "execute, test and debug ... software designed for a new hardware architecture" – his whole purpose would be defeated if the new architecture were modified as taught in Goetz '913. Similarly, Murphy '947 is directed to providing a migration tool for VAX software onto Alpha hardware for customer software[4] – that purpose would be defeated if the VAX were modified in the manner taught by Goetz'.
2. Because Goetz '913 has already "ironed out" all of the differences between the two architectures, he provides no mechanism by which software could assist in a cross-architecture switch. Goetz '913, col. 17, lines 41-44 teaches that all that is required on an architecture switch is wait a few cycles for the pipeline to drain, "without exceptions" or other control transfer that might allow software to gain control. In contrast, Brender '422 and Murphy '947 exhaustively analyze every possible case to make sure that a software jacket is invoked on every switch between architectures.
3. Goetz '913 is implemented essentially entirely in hardware, while Brender '422 and Murphy '947 are implemented entirely in software.

Difference no. 1 defeats "motivation to combine" – the "principle of operation" of one or the other would have to be abandoned in order to combine Goetz '913 with either Brender '422

---

[3] In many cases, the data structures will have identical values, and may be collapsed together. Nonetheless, both Brender '422 and Murphy '947 start by analyzing each conceivable transfer as a particular case.

[4] *See* Murphy '947 at col. 2, line 50-55; Anton Chernoff, et al., FX!32, A Profile-Directed Binary Translator, IEEE Micro, Vol. 18 No. 2 pp. 56-64 (March/April, 1998); Raymond J. Hookway, Mark A. Herdeg, Digital FX!32: Combining Emulation and Binary Translation (August 28, 1997), http://www.digital.com/info/DTJP01HM.HTM; Digital Equipment Corp., White Paper: How DIGITAL FX!32 Works (September 1997) http://www.digital.com/semiconductor/amt/fx32/fx-white.htm, all three of which are of record in this application.

or Murphy '947. MPEP § 2143.01 disallows an obviousness combination in such circumstances ("If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then ... the references are not sufficient to render the claims *prima facie* obvious.")

Difference no. 2 defeats "reasonable expectation of success:" Brender's or Murphy's software cannot work on Goetz' machine because Goetz expressly states that his switch occurs "without exceptions" or other mechanism that might allow software to be invoked on a switch. Goetz '913, col. 17, lines 41-44 (switch occurs "without exceptions" triggered by the switch itself).

Difference no. 3 also defeats "reasonable expectation of success." Goetz '913 is implemented essentially entirely in hardware. When setting up a fab line for a chip design takes several weeks and several million dollars, no reasonable person in the art would even consider implementing Brender's and Murphy's data structures, which are specialized to one single program, in hardware. That enormous cost would have to be borne for each version of each program. Further, when hardware costs rise with the cube of die size, no sane person would consider implementing Brender's and Murphy's data structures, one specifically created for each conceivable transition, in hardware.

Fourth, neither Goetz '913, Brender '422 or Murphy '947 teach any technique that would bridge these differences. One confronted with these three references (assuming for the moment that "motivation" to pick these three existed), in order to make them work together, would have to remove several crucial features of the prior art, and design substantial "glue" hardware and/or software that is taught in no reference.

Paragraph 14.6 of the October 2004 Office Action attempts to justify the combination by showing that "software and hardware are logically equivalent and interchangeable." This paragraph of the Action misstates the law, and misstates the engineering realities.

First, paragraph 14.6 is no more than a statement that "references <u>can</u> be combined or modified," reasoning that is forbidden at MPEP § 2143.01. No Office Action has ever shown <u>motivation</u> to combine these references.

Second, Brender's and Murphy's software techniques cannot "reasonably" be implemented in hardware: both Brender '422 and Murphy '947 require frequent modification that is not practical in hardware. Brender '422 teaches that an entire program must be compiled and linked, if any single instruction changes, on the same basis as any other program. Brender '422, col. 7-16, see esp. col. 15, lines 24-57 (both "manual" and "automatic" jacketing require modification of the binary image at "image build time"); Murphy '947, col. 7, lines 6-15. Hardware cannot be generated on such a "throw-away" basis in any cost-effective manner – for example, any such attempt would require several weeks of turn-around time and (typically) hundreds of thousands of dollars. Instead, hardware is designed to be general, so that it need not be re-fabricated for each revision. Accordingly, Applicants' specification discloses a few simple, general structures and techniques that can be adapted to allow cross-instruction-set calls in many different programs. See, e.g., sections I-IV of the specification, pages 31-75.

Third, a key difference between hardware and software is the cost curve. The cost of software goes up roughly linearly with capability. In contrast, the cost of hardware goes up roughly as the cube of die size, or roughly as the cube of chip capability. See lecture slides from the University of California at Berkeley, available at vco.ett.utu.fi/courses/ETT_2015/kalvot/luento4.pdf, excerpts attached hereto as Exhibit A. Therefore, most CPU chip designs avoid large on-chip structures that are not directly related to the core tasks of executing instructions. Because Brender '422 and Murphy '947 are entirely software based, they can use large structures that were not amenable to hardware implementation at the time of the invention. For a first example, Brender '422 and Murphy '947 teach that an entire program is linked together for execution. Brender '422, col. 15, lines 24-57; Murphy '947, col. 7, lines 6-40. Implementing an entire program in hardware would not be recognized as desirable by one of ordinary skill. For a second example, Murphy's and Brender's "jacketing tables" and "jacketing routines" may be quite large, because a specific data structure must be generated for each routine that could conceivably be called from the opposite instruction set. E.g., Murphy '947, col. 6, lines 4-8 ("an array that forms a signature ... for each incoming call in each Y code routine..."); Brender '422, col. 8, lines 31-37 ("Jacketing requires that knowledge ... for each subprogram in each domain..."); Brender '422, col. 10, line 43 through col. 15, line 16. Because Brender '422

and Murphy '947 use entirely-software-based approaches, they have no motivation to teach techniques that could practically be implemented in hardware. Implementing their large, non-general structures in hardware would be neither reasonably practical ("reasonable expectation of success") nor desirable ("motivation to combine") to one of ordinary skill.

All rejections based on combinations of Goetz '913 with either Brender '422 or Murphy '947 may be withdrawn.

VI. Claim 94

Claim 94 is discussed at paragraphs 43-44 of the Office Action of February 2004, and in paragraph 14.5 of the October 2004 Action, in the context of Brender '422 alone (with Murphy '947 incorporated by reference). The February 2005 Advisory Action makes no response to the substantive arguments on claim 94 made in Applicant's papers. Claim 94 recites as follows:

> 94. A method, comprising the steps of:
>
> executing a section of computer object code twice, without modification of the code section between the two executions, the code section materializing a destination address into a register and being architecturally defined to directly transfer control indirectly through the register to the destination address, the two executions materializing two different destination addresses;
>
> the two destination code sections at the two materialized destination addresses being coded in two distinct instruction sets and, respectively, obeying the default calling conventions native to each of the two instruction sets, neither instruction set being a subset of the others.

Even taken together, the discussion in the two Office Actions is too incomplete to constitute a rejection. For example, claim 94 recites a "register" that performs certain functions. The Office Actions pointedly ignore this word of claim 94, and fail to indicate any "register" that is thought to perform these functions. No rejection exists.

As noted in § V.B at page 41 above, the Examiner's technological analysis of Brender '422, col. 21, lines 50-57 is incorrect. In fact, the paragraph following the second paragraph cited in the Office Action, Brender '422 does discuss a "register," and states that it is not the "register" of claim 94:

> The only change is to replace the load of R26 from the second quadword of the target procedure descriptor with a load of the code address for [the runtime routine that implements cross-ISA calls]. The code address for the [runtime

routine] is loaded as opposed to a procedure value address; [the runtime routine] is invoked with a non-standard call and there is no associated procedure descriptor.

Brender '422 makes clear that the "destination address" materialized into register R26 is "always" the address of the special jacketing runtime routine, not a "direct" destination.

Claim 94 is neither rejected nor rejectable over Brender '422.

VII. Claim 96

Claim 96 is briefly mentioned, without being rejected, in paragraph 100 of the Office Action of February 2004, over unspecified portions of Goetz '913, Brender '422, and Murphy '947, combined in some unspecified way. A single limitation of claim 96 is discussed in paragraph 14.4 of the October 2004 Action. The Advisory Action of February 2005 is silent the substantive arguments on claim 96 made in Applicant's papers. Claim 96 recites as follows:

Claim 96 recites as follows:

96. A microprocessor chip, comprising:

two instruction decoders designed to decode instructions of first and second instruction sets, respectively, and circuitry of a single instruction pipeline designed to execute the instructions decoded by either of the two instruction decoders;

circuitry and/or software designed to detect when execution flows or transfers control from code coded in one instruction set to code coded in the other, program code in the first and second instruction sets using first and second different data storage conventions, respectively; and

circuitry and/or software designed to respond to the detection by altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

Claim 96 is patentable because Goetz '913 is not combinable with Brender '422 and Murphy '947, as discussed in § V.C at page 42, above.

VIII. Claim 104

Claim 104 was briefly mentioned in paragraphs 10 and 42 of the Office Action of February 2004 in the context of Goetz '913 alone. Because the Office Action of February 2004 made no element-by-element comparison of claim 104 to Goetz '913 or any other reference, no rejection was raised in February 2004.

Against Applicant's better judgment (see § IX at page 49, above), Applicant will make a good faith guess at the Examiner's view, and respond. Claim 104 recites as follows:

> 104. A method, comprising the steps of:
>
> executing instructions fetched from first, second and third regions of a single address space of the memory of a computer, the instructions of the first region being coded for execution by computers of a first architecture and following a first data storage convention, the instructions of the second region being coded for execution by computers of a second architecture and following the first data storage convention, the instructions of the third region being coded for execution by computers of the second architecture and following a second data storage convention, the memory regions having associated modifiable indicator elements, a hardware structure for storing the indicator elements enforcing a requirement that the memory regions be necessarily disjoint, the modifiable indicator elements each having a value indicating the architecture and data storage convention under which instructions from the associated region are to be executed;
>
> when execution of the instruction data flows or transfers between the first, second and third regions, adapting the computer for execution in the architecture and/or convention of the region transferred to.
>
> when execution of the instruction data flows or transfers between the first, second and third regions, adapting the computer for execution in the architecture and/or convention of the region transferred to.

Claim 104 recites three memory regions – a first and third in which the instruction architecture and data storage convention are matched to each other, and a second in which the instruction architecture matches the first region, and the data storage convention matches the third region.

Goetz '913 is careful to always maintain one or the other. Goetz '913 discloses nothing analogous the second region of claim 104. Accordingly, claim 104 is not anticipated by Goetz '913.

IX. Claim 22

Claim 22 is discussed at ¶ 52 of the Office Action of the February 2004, ¶ 14 of the October 2004 Office Action, and ¶ 4 of the February 2005 Advisory Action. Claim 22 recites as follows:

> 22. A method, comprising the steps of:
>
> executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed;
>
> recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

The Examiner has not set down on paper any discernable position with respect to claim 22, let alone in the manner required by 37 C.F.R. § 1.104, or timely as required by § 1.113. 37 C.F.R. § 1.104(c)(2) reads as follows:

> **§ 1.104 Nature of examination.**
>
> (2) In rejecting claims for want of novelty or for obviousness, the examiner must cite the best references at his or her command. When a reference is complex or shows or describes inventions other than that claimed by the applicant, the particular part relied on must be designated as nearly as practicable. The pertinence of each reference, if not apparent, must be clearly explained and each rejected claim specified.

Giving ¶¶ 52.3 and 52.4 of the February 2004 Action their most generous reading, the Office Action compared the "the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed" to only Goetz '913, Fig. 10, col. 14, lines 18-22, and col. 17, lines 24-33 and to no other reference.

These portions off Goetz '913 discuss no fewer than 20 different data elements and signals. In such situations, every other examiner in the Office complies with Rule 104(c)(2)'s requirement for a "clear explanation" by including the names of particular elements from the reference, along with the designation of portions of the references. Examiner Ellis is the rare – perhaps only – exception.

Nonetheless, in a good faith effort to advance prosecution, this attorney went to great effort to discern the interpretation of the references most likely intended by the Examiner. That

interpretation was set forth in the Response to Office Action of July 2004. The July 2004 fully responded to all issues fairly discernable in the February 2004 Office Action.

The Office Action of October 2004 hints that the Examiner believes that other references are pertinent to the particular claim language addressed in ¶¶ 52.3 and 52.4. The October Office Action does not even specify which other references might be involved, let alone "designate the portions relied on" or "clearly explain" their pertinence. The October Office Action responds to none of the substantive arguments raised in Applicant's paper of July 2004.

The February 2005 Advisory Action further confuses the issues: it merely states that unspecified portions of one or both of two other references might be used to supplement the Goetz '913 patent, in some unspecified way, to meet this claim language. The February 2005 advisory Action is totally silent on the substantive arguments of both the Response to Office Action of July 2004 (§ II, at pages 34-35) and the Response to Office Action of January 25, 2005 (§ II, at pages 35-36).

It is now clear that it is simply futile to attempt to read the Examiner's mind. Often, the Examiner's hidden position has been far too clever and speculative for any reasonable applicant to discern – for example, that the term "necessarily disjoint" covers "may overlap" because the word "necessarily" introduces some possibility of an exception,[5] or that reversing the order of the bytes of a number would leave the value of the number unaffected.[6] Examiner Ellis has "discovered" exceptions to a number of MPEP provisions that exist nowhere in written form.[7] No Applicant could have guessed at these positions of the Examiner, before they were written down. Applicant's attempts to respond in good faith to the Examiner's bare-bones allegations, and requests that the Examiner comply with the rules established by the Director, have not been reciprocated.

Applicant respectfully reminds the Examiner of the instructions issued by the Director:

---

[5] See discussion of claim 104 in Advisory Action of March 27, 2003.

[6] Compare Advisory Action of April 14, 2004, the first time this position is articulated, to Declaration of David R. Levine, April 10, 2003.

[7] For example, in the paper of February 10, 2003, Examiner Ellis states that he believes that "reasonable expectation of success" only applies in the chemical arts, not "all arts" as stated in MPEP § 2142.

1. Every written rejection must both "designate" particular portions of each reference relied on and "clearly explain" the pertinence, *e.g.*, by identifying particular elements of the references by name.
2. Every obviousness rejection must make the three *prima facie* showings set forth in MPEP § 2143-2143.03. MPEP § 2142 ("The legal concept of *prima facie* obviousness is a procedural tool of examination which applies broadly to all arts. ... The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness").
3. An Office Action must "Answer All Material Traversed." MPEP § 707.07(f).

Applicant has responded to every issue set on paper by the Examiner, and has made a good faith attempt to guess at those points where the Examiner has been silent. See Applicant's papers of January 2005 and July 2004. Until the Examiner puts his reasoning on paper in the manner instructed by the Director, and responds to Applicant's papers, (a) as a matter of law, the claims are not rejected, and (b) Applicant cannot meaningfully respond, let alone identify any difference of opinion for appeal.

X. Dependent claims

The dependent claims are patentable with the independent claims discussed above. In addition, the dependent claims recite additional features that further distinguish the art.

XI. The IDS of July 15, 2004

Paragraph 4 of the February Advisory Action states that many references cited in the IDS of July 2004 were not considered because no legible copies were provided. The Advisory Action misstates the law. 37 C.F.R. § 1.98 provides that references will be considered if they are in the file of any application from which § 120 priority is claimed. Legible copies of each reference of which copies were required were filed in application serial no. 09/239,194, from which priority is claimed under § 120. Applicant requests consideration to these references.

A number of U.S. patent references were refused consideration, for no stated reason. Applicant notes that the rules in effect at the time required neither copies of these references nor a Rule 98(d) statement. Applicant requests consideration of these references.

A replacement IDS accompanies the copy of this paper submitted by First Class Mail.

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. A Petition for Extension of Time for 3 months is included. In the event that any further extension of time is required, Applicant petitions for that extension of time required to make this response timely. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-03-4000.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: April 14, 2005

By: _____

David E. Boundy
Registration No. 36,461

WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8757
(212) 728-9757 Fax

Exhibit A to

Second Response to Office Action

CS152
Computer Architecture and Engineering

Lecture 5: Cost and Design

September 10, 1997

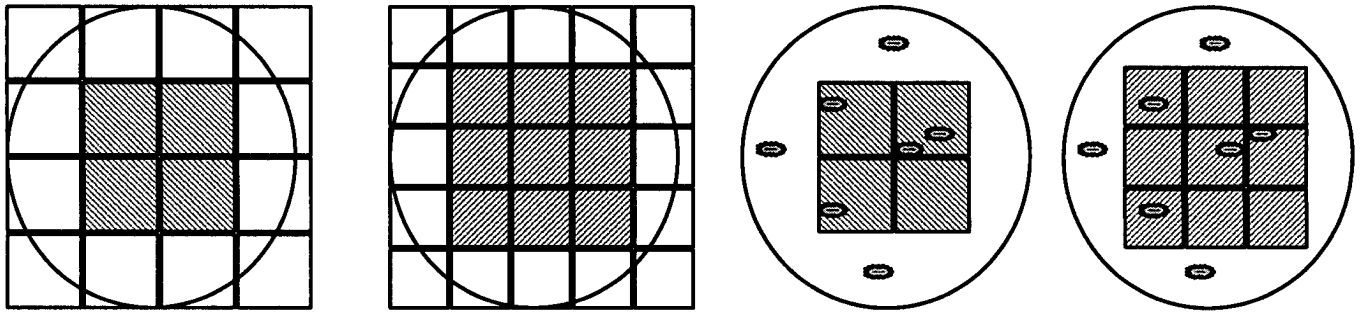
Dave Patterson (http.cs.berkeley.edu/~patterson)

lecture slides: http://www-inst.eecs.berkeley.edu/~cs152/

Integrated Circuit Costs

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} * \text{Die yield}}$$

$$\text{Dies per wafer} \sim \text{eff} \frac{\text{Wafer Area}}{\text{Die Area}}$$



$$\text{Die Yield} = \frac{\text{Wafer yield}}{\left\{ 1 + \frac{\text{Defects\_per\_unit\_area} * \text{Die\_Area}}{?} \right\}^{?}}$$

Die Cost is goes roughly with the cube of the area.

Die Yield

Raw Dices Per Wafer

wafer diameter	*die area (mm^2)*					
	100	*144*	*196*	*256*	*324*	*400*
6"/15cm	139	90	62	44	32	23
8"/20cm	265	177	124	90	68	52
10"/25cm	431	290	206	153	116	90
die yield	23%	19%	16%	12%	11%	10%

typical CMOS process: ? =2, wafer yield=90%, defect density=2/cm2, 4 test sites/wafer

Good Dices Per Wafer (Before Testing!)

6"/15cm	31	16	9	5	3	2
8"/20cm	59	32	19	11	7	5
10"/25cm	96	53	32	20	13	9

typical cost of an 8", 4 metal layers, 0.5um CMOS wafer: ~$2000

Real World Examples

Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX2	3	0.80	$1200	1.0	81	181	54%	$12
PowerPC 601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA 7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
SuperSPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Other Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

Packaging Cost: depends on pins, heat dissipation

Chip	*Die cost*	*Package pins*	*Package type*	*Package cost*	*Test & Assembly*	*Total*
386DX	$4	132	QFP	$1	$4	$9
486DX2	$12	168	PGA	$11	$12	$35
PowerPC 601	$53	304	QFP	$3	$21	$77
HP PA 7100	$73	504	PGA	$35	$16	$124
DEC Alpha	$149	431	PGA	$30	$23	$202
SuperSPARC	$272	293	PGA	$20	$34	$326
Pentium	$417	273	PGA	$19	$37	$473